TASK	VJEKO	FONT	KOCKICE	KRUŽNICE	HASH	GRAŠKRIŽJA

source code	`vjeko.pas`
	`vjeko.c`
	`vjeko.cpp`
		`font.pas`
		`font.c`
		`font.cpp`
			`kockice.pas`
			`kockice.c`
			`kockice.cpp`
				`kruznice.pas`
				`kruznice.c`
				`kruznice.cpp`
					`hash.pas`
					`hash.c`
					`hash.cpp`
						`graskrizja.pas`
						`graskrizja.c`
						`graskrizja.cpp`

input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	1 second	1 second	1 second
memory limit	32 MB	32 MB	32 MB	32 MB	256 MB	32 MB
point value	**50**	**80**	**100**	**120**	**140**	**160**
	650					

Problems translated from Croatian by: **Paula Gombar**

In his spare time, Vjeko likes to browse through files in directories. Unfortunately, it seems to him that the console on his computer broke down and now it doesn't correctly print file names that match a certain pattern.

A pattern is string consisting of **lowercase letters of the English alphabet** and **exactly one** asterisk. A file name matches a pattern if the pattern string can be made equal to the file name by replacing the asterisk with an **arbitrary** string of lowercase letters of the English alphabet (an empty string substitution is also possible). For example, strings "abcd", "ad" and "anestonestod" all match the pattern "a*d" and the string "bcd" does not.

Write a programme that will, given a pattern and file names, output whether a file name matches the pattern or not.

INPUT

The first line of input contains the integer **N** ($1 \leq \mathbf{N} \leq 100$), the number of files.

The second line of input contains a string of characters consisting of only lowercase letter of the English alphabet and exactly one asterisk (ASCII value 42). The length of the string will not exceed 100 and the asterisk will not be located at the beginning nor at the end of the string.

Each of the following **N** lines contains file names, each in its own line. The file names consist of only lowercase letters of the English alphabet and their length will not exceed 100.

OUTPUT

Output **N** lines. The **i**th line should be "DA" (Croatian for yes) if the **i**th file name matches the pattern or "NE" (Croatian for no) if the **i**th file name does not match the pattern.

SAMPLE TESTS

input

```
3
a*d
abcd
anestonestod
facebook
```

output

```
DA
DA
NE
```

input

```
6
h*n
huhovdjestvarnomozedocisvastan
honijezakon
atila
je
bio
hun
```

output

```
DA
DA
NE
NE
NE
DA
```

Little Ivica got himself a summer job at a company that produces computer fonts. The branch of the company where Ivica works at specialises in testing computer fonts and Ivica's team is responsible of testing **only lowercase letters of the English alphabet**.

The letters are tested so that various sentences using those letters are printed out and then manually (more accurately, visually) checked if everything is arranged properly. Only sentences which contain **all 26 lowercase letter of the English alphabet (a-z)** are used for testing purposes. These sentences are called **test sentences**.

You've probably already assumed that Ivica's job is to find test sentences. He has a dictionary which consists of **N** words and has to calculate how many different test sentences can be made out of those words. Every word from the dictionary can be used **only once** in the sentence and the order of the words in the sentence is **irrelevant** (i.e. "uvijek jedem sarmu" and "jedem sarmu uvijek" are equal sentences).

INPUT

The first line of input contains the integer **N** ($1 \leq \mathbf{N} \leq 25$), the number of words in the dictionary.

Each of the following **N** lines contains a single word from the dictionary, its length not exceeding 100. All the words from the dictionary are going to be **unique**.

OUTPUT

The first and only line of output must contain the required number from the task.

SAMPLE TESTS

input	**input**	**input**
9 the quick brown fox jumps over a sleazy dog	3 a b c	15 abcdefghijkl bcdefghijklm cdefghijklmn defghijklmno efghijklmnop fghijklmnopq ghijklmnopqr hijklmnopqrs ijklmnopqrst jklmnopqrstu klmnopqrstuv lmnopqrstuvw mnopqrstuvwx nopqrstuvwxy opqrstuvwxyz
output	**output**	**output**
2	0	8189

Clarification of the first example: All words but the word "a" must be used in the test sentence because each word contains a letter that cannot be found in any other word. Therefore, there are two possible solutions. The first one being the sentence which consists of all the words and the second one being the sentence which consists of all the words apart from the word "a".

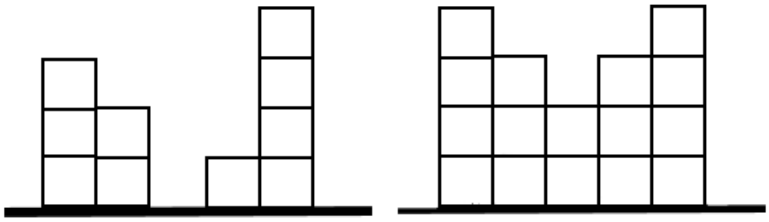
Clarification of the third example: Given that the example is long, we stress that all the words from the example consist of consecutive letters of the English alphabet.

Mirko and Slavko are playing with bricks. Both of them have their own pile of bricks. The piles consist of **N** columns (where **N** is an odd number). The number of bricks in the **i**th column of Mirko's pile is labeled with $\mathbf{m}_i$ and Slavko's pile with $\mathbf{s}_i$.

They have decided to create two **equal** piles constructed in a way that the heights of columns are **strictly descending** at first and then **strictly ascending** (see right image below) and the heights of adjacent columns differ **exactly** by 1 (see image). The lowest of the columns must have an **equal** number of columns to the left and to the right of it.

The piles can be modified by removing **one** brick from **the top of some column** and throw it out the window (they **cannot reuse** it) or by taking **one** brick from the box and place it on **the top of some column** (there is an infinite amount of bricks in the box). Removing or placing a brick counts as one move.

You have to determine the **minimal** number of moves so that Mirko and Slavko can rearrange their piles in the described way.



On the left, there is a pile with column heights 3, 2, 0, 1 and 4.
On the right, there is one of the possible final layouts.

INPUT

The first line of input contains an **odd** number **N**. ($1 \leq \mathbf{N} \leq 300\,000$), the number of columns in both piles.

The second line of input contains **N** integers $\mathbf{m}_i$ ($0 \leq \mathbf{m}_i \leq 10^{12}$), column heights in Mirko's pile.

The third line of input contains **N** integers $\mathbf{s}_i$ ($0 \leq \mathbf{s}_i \leq 10^{12}$), column heights in Slavko's pile.

OUTPUT

The first and only line of output must contain the minimal number of moves.

SCORING

In test cases worth 40% of total points, the following will hold: $1 \leq \mathbf{N} \leq 1\,000$ i $0 \leq \mathbf{m}_i, \mathbf{s}_i \leq 1\,000$.

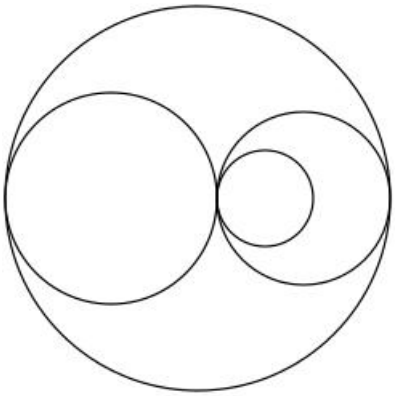
SAMPLE TESTS

input	input
3 1 2 3 3 2 2	5 2 3 0 1 4 3 3 2 3 1
output	**output**
3	10

Clarification of the first example: Mirko places two bricks on the top the first column in his pile and Slavko places one brick on the top of the third column in his pile.

Enjoying a casual afternoon walk in the coordinate system, little Luka has encountered **N unique** circles with its centers lying on the x-axis. The circles **do not intersect**, but they **can touch** (from the inside and the outside). Fascinated with circles, Luka wondered how many **regions** the circles divide the plane into. Of course, you are going to help him answer this question.

A **region** is a set of points such that each two points can be connected with a **continuous** curve, without cutting through any of the circles.



One of the possible layouts of circles

INPUT

The first line of input contains the integer **N** ($1 \leq \mathbf{N} \leq 300\,000$), the number of circles.

Each of the following **N** lines contains two integers $\mathbf{x_i}$ and $\mathbf{r_i}$ ($-10^9 \leq \mathbf{x_i} \leq 10^9$, $1 \leq \mathbf{r_i} \leq 10^9$), the number $\mathbf{x_i}$ representing the x coordinate of the $\mathbf{i^{th}}$ circle and the number $\mathbf{r_i}$ representing the radius of the $\mathbf{i^{th}}$ circle.

All the circles in the input will be unique.

OUTPUT

The first and only line of output must contain the required number from the task.

SCORING

In test cases worth 40% of total points, the **N** will not exceed 5 000.

SAMPLE TESTS

input	**input**	**input**
2 1 3 5 1	3 2 2 1 1 3 1	4 7 5 -9 11 11 9 0 20
output	**output**	**output**
3	5	6

Clarification of the third example: The example corresponds to the image in the task statement.

Little Mirko is studying the *hash* function which associates numerical values to words. The function is defined recursively in the following way:

- f(empty word) = 0
- f(word + letter) = ((f(word) * 33) XOR ord(letter)) % MOD

The function is defined for words that consist of only lowercase letters of the English alphabet. XOR stands for the bitwise XOR operator (i.e. 0110 XOR 1010 = 1100), ord(letter) stands for the ordinal number of the letter in the alphabet (ord(a) = 1, ord(z) = 26) and A % B stands for the remainder of the number A when performing integer division with the number B. MOD will be an integer of the form $2^{\mathbf{M}}$.

Some values of the hash function when **M** = 10:

- f(a) = 1
- f (aa) = 32
- f (kit) = 438

Mirko wants to find out how many words of the length **N** there are with the *hash* value **K**. Write a programme to help him calculate this number.

INPUT

The first line of input contains three integers **N**, **K** and **M** ($1 \leq \mathbf{N} \leq 10$, $0 \leq \mathbf{K} < 2^{\mathbf{M}}$, $6 \leq \mathbf{M} \leq 25$).

OUTPUT

The first and only line of output must consist of the required number from the task.

SCORING

In test cases worth 30% of total points, **N** will not exceed 5.

Additionally, in test cases worth 60% of total points, **M** will not exceed 15.

SAMPLE TESTS

input	**input**	**input**
1 0 10	1 2 10	3 16 10
output	**output**	**output**
0	1	4

Clarification of the first example: None of the characters in the alphabet has an ord value 0.

Clarification of the second example: It is the word "b".

Clarification of the third example: Those are the words "dxl", "hph", "lxd" and "xpx".

Peatown has become a metropolis. We can observe it as a rectangular grid of streets. There are fifty thousand vertical streets running north-south (labeled with x-coordinates from 1 to 50 000) and fify thousand horizontal streets running east-west (labeled with y-coordinates from 1 to 50 000). All streets are two-way streets. An intersection of a horizontal and vertical street is called a crossroads.

Residents of Peatown are very irresponsible and reckless. They drive like idiots so the mayor of Peatown has decided to place traffic lights on **N** crossroads. A **path** between two crossroads is **dangerous** if there is **a turn without a traffic light**. Otherwise it is harmless.

It is not possible to ensure that all paths are harmless, but the mayor of Peatown is satisfied if **between each two traffic lights** at least **one of the shortest paths** is **harmless**. Unfortunately, the current distribution of traffic lights is too dangerous. Your task is to place **additional traffic lights** (less than 700 000 of them) so that the set of traffic lights (which contains both new and old traffic lights) meets the mayor's requirement. Surely you're not pea-brained so help the residents of Peatown!

INPUT

The first line of input consists of an integer **N** ($2 \leq \mathbf{N} \leq 50\,000$), the number of initially placed traffic lights.

Each of the following **N** lines contains a location of one traffic light, represented with integers **X** and **Y** ($1 \leq \mathbf{X}, \mathbf{Y} \leq 50\,000$), coordinates of the vertical and horizontal streets which intersect in that crossroads. All traffic lights will be unique.

OUTPUT

Output the locations of new traffic lights, each in its own line.

Placing multiple traffic lights on the same location is allowed.

The number of new traffic lights must be **smaller than 700 000**.

SAMPLE TESTS

input	**input**	**input**
2 1 1 3 3	3 2 5 5 2 3 3	5 1 3 2 5 3 4 4 1 5 2
output	**output**	**output**
1 3	3 5 5 3	1 5 3 3 3 5 4 2 4 3